

REMARKS

Applicant amended independent claim 13 to include a feature, similar to the feature recited in claims 19, that one or more additional instructions are executed after execution of the instruction that issued the request to the shared resource. Applicant cancelled claim 19. Applicant similarly amended independent claims 26 and 31. After these amendments, claims 13-18 and 20-35 are pending. Claims 13, 26 and 31 are independent.

The examiner rejected claim 22 under 35 U.S.C. §112, first paragraph, on the grounds that “[c]laim 22 claims an instruction that identifies a signal generated in response to the request for a shared resource. However this feature has not been described in the specification therefore claim 22 is rejected as failing to comply with written description requirement” (Office Action, Page 2).

Applicant respectfully disagrees with the examiner's contention.

Applicant's originally filed application describes, at page 22 (immediately after the heading Context Switch), “Referring to FIG. 3B, a format from a context switch instruction is shown. A context switch is a special form of a branch that causes a different context (and associated PC) to be selected.” FIG. 3B, in turn, shows that the context switch instruction includes, at bits 8-15 the wake-up event (i.e., signals) that would cause the swapped-out context to be swapped back in. As shown, some of those signal include the SRAM signal, the FBI signal, etc.

Further, the last paragraph of page 13 and the first paragraph of page 14 of the originally filed application describe:

The microengine 22f also includes context event switching logic 74. Context event logic 74 receives messages (e.g., SEQ_#_EVENT_RESPONSE; FBI_EVENT_RESPONSE; SRAM_EVENT_RESPONSE; SDRAM_EVENT_RESPONSE; and ASB_EVENT_RESPONSE) from each one of the shared resources, e.g., SRAM 26a, SDRAM 26b, or processor core 20, control and status registers, and so forth. These messages provide information on whether a requested function has completed. Based on whether or not a function requested by a thread has completed and signaled completion, the thread needs to wait for that completion signal, and if the thread is enabled to operate, then the thread is placed on an available thread list (not shown). The microengine 22f can have a maximum of e.g., 4 threads available.

Thus, the above-referenced paragraphs describe that context-switching can be performed by executing a context-switch instruction, and that in the course of performing context-switching, signals are used (e.g., signals that indicate that functions requested by a thread have been completed). Accordingly, applicant's specification provides ample written support to the feature recited in claim 22.

Applicant, therefore, respectfully submits that claim 22 complies with the written description requirement under 35 U.S.C. §112, first paragraph.

The examiner rejected claims 13-21 and 24 under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 6,295,600 to Parady in view of U.S. Patent No. 6,507,862 to Joy. The examiner further rejected claims 22 and 23 under 35 U.S.C. §103(a) as being unpatentable over Parady in view of Joy. Additionally, the examiner rejected claim 25 under 35 U.S.C. §103(a) as being unpatentable over Parady in view of Joy, and further in view of U.S. Patent No. 6,085,215 to Ramakrishnan. The Examiner also rejected claims 26-35 under 35 U.S.C. §103(a) as being unpatentable over Ramakrishnan, in view of Parady and Joy.

Specifically, with respect to applicant's feature pertaining to executing additional instructions after execution of the instruction that issued the memory request, which was recited in claim 19 and now appears in amended independent claim 13, the examiner stated:

12. As per claim 19, Parady taught executing additional instructions of the first thread after the at least one instruction to issue the request to the shared resource and before swapping the first thread out[in one embodiment, on a non-blocking loads the system supports allowing the program to continue in the same program thread which memory access is being completed] (e.g. see col. 4, lines 63-66). (Office Action, Page 5)

Applicant respectfully disagrees with the examiner's characterization of Parady.

Applicant's independent claim 13 recites "[a] method, comprising: ... executing at least one instruction of a first thread having a first program counter, the at least one instruction including at least one instruction to issue a request to a resource shared by the multiple processing engines; executing one or more additional instructions of the first thread, after executing the at least one instruction to issue the request to the shared resource."

As explained in applicant's Amendment in Reply to Action of November 18, 2005, Parady describes apparatus for switching between threads of a program in response to a long-latency event (Abstract). Parady further describes:

In one embodiment, the present invention also supports non-blocking loads which allow the program to continue in the same program thread while the memory access is being completed. Preferably, such non-blocking loads would be supported in addition to blocking loads, which stall the operation of the program thread while the memory access is being completed. Thus, there would not be a thread switch immediately on a non-blocking load, but would be upon becoming a blocking load waiting for data (or store or other long-latency event). (col. 4, line 63 to col. 5, line 5)

Thus, Parady's apparatus supports non-blocking load operations that cause the apparatus to continue executing the same thread even though the thread involves what potentially could be a lengthy memory access operation. Under those circumstances, execution of the thread that caused the non-blocking load operation will continue until the thread encounters a blocking load operation. At that point, the encountered blocking load operation causes the apparatus to switch to another thread while the memory access for the blocking load operation is being completed. But the non-blocking operation that preceded the blocking load operation does not itself cause thread switching.

Causing a thread to continue execution after performing a non-blocking operation is entirely different from causing a first thread to continue executing for one or more additional instructions before swapping execution of the first the thread with another thread. In the latter situation, swapping does occur, it just does not occur immediately. Thus, contrary to the examiner's contention, Parady neither discloses nor suggests at least the feature of "executing one or more additional instructions of the first thread, after executing of the at least one instruction to issue the request to the shared resource," as required by applicant's independent claim 13.

Joy describes a multi-threading processor architecture that includes multiple vertically-threaded processors (see for example, FIG. 9, and accompanying description at col. 18, lines 39 to col. 20, line 63.) While Joy describes that thread switching is implemented using high speed multiple flip-flops (see FIG. 4 and accompanying description at col. 10, line 41 to col. 13, line 7), and a thread switch logic that at least in part controls the flip-flops, Joy does not describe the

functionality of continuing execution of a thread for one or more additional instructions after execution of an instruction that issued a memory request and before thread swapping is performed. Thus, Joy neither discloses nor suggests at least the feature of "executing one or more additional instructions of the first thread, after executing of the at least one instruction to issue the request to the shared resource," as required by applicant's independent claim 13.

Because neither Parady, nor Joy, discloses or suggests, alone or in combination, at least the feature of "executing one or more additional instructions of the first thread, after executing of the at least one instruction to issue the request to the shared resource," applicant's independent claim 13 is patentable over the cited art.

Claims 14-18 and 20-25 depend from independent claim 13, and are therefore patentable for at least the same reasons as independent claim 13.

As noted, the examiner rejected claims 26-35 under 35 U.S.C. §103(a) as being unpatentable over Ramakrishnan, in view of Parady and Joy.

Amended independent claims 26 and 35 recite "multiple, multi-threaded processing engines, individual ones of the engines including an arbiter to select a thread to execute, the multiple multi-threaded processing engines configured to execute one or more instructions of a first thread after execution of an instruction of the first thread that issued a request to a shared resource, and before swapping of the first thread for another thread," or similar language.

For reasons similar to those provided with respect to independent claim 13, at least this feature is not disclosed or suggested by Parady and/or Joy.

Ramakrishnan describes method and apparatus for avoiding receive livelock and transmit starvation, and for minimizing packet loss and latency in a communication network station (Abstract). To that end, Ramakrishnan explains that "[b]riefly, and in general terms, the method of the invention comprises the steps of dividing processing tasks into processing threads, each of which is structured to execute for a limited time before being subject to preemption by another processing thread" (col. 4, lines 16-20). In particular, Ramakrishnan's apparatus uses a scheduler to control execution of the thread. Ramakrishnan explains:

The core thread scheduler 46 is a special task to which control is transferred at the end of each processing thread. The scheduler is simply a round-robin selection device with knowledge of all of the processing threads and access to the real time thread flags 50. When one processing thread is

completed, the scheduler 46 chooses the next one in turn having a flag set to indicate that processing is needed. FIG. 5 shows the basic steps performed by the core thread scheduler 46. After return from a completed processing thread, indicated at 56, the scheduler 46 selects the next thread that needs to run (block 58), and yields control to the next thread (block 60). The scheduler 46 includes at least one time slot in its round robin system for the general purpose domain 42. (col. 9, lines 9-22)

So while Ramakrishnan divides tasks to processing threads which are selected by the scheduler, no where does Ramakrishnan describe that before selection of a different thread, one or more instructions, of a currently executing thread that issued a memory request instruction, are executed. Indeed, since processing threads on Ramakrishnan's apparatus are selected only after completion of the preceding thread, the preceding thread has no remaining instructions that need to be executed. Thus, Ramakrishnan does not disclose or suggest at least the feature of "the multiple multi-threaded processing engines configured to execute one or more instructions of a first thread after execution of an instruction of the first thread that issued a request to a shared resource, and before swapping of the first thread for another thread," as required by applicant's independent claims 26 and/or 31.

Since none of Parady, Joy, and Ramakrishnan, discloses or suggests, alone or in combination, at least the feature of "the multiple multi-threaded processing engines configured to execute one or more instructions of a first thread after execution of an instruction of the first thread that issued a request to a shared resource, and before swapping of the first thread for another thread," applicant's independent claims 26 and 31 are patentable over the cited art.

Claims 27-30 depend from independent claim 26 and are therefore patentable for at least the same reasons as independent claim 26. Claims 32-35 depend from independent claim 31 and are therefore patentable for at least the same reasons as independent claim 31.

It is believed that all the rejections and/or objections raised by the examiner have been addressed.

In view of the foregoing, applicant respectfully submits that the application is in condition for allowance and such action is respectfully requested at the examiner's earliest convenience.

Canceled claims, if any, have been canceled without prejudice or disclaimer.

Applicant : Debra Bernstein et al.
Serial No. : 10/643,438
Filed : August 19, 2003
Page : 12 of 12

Attorney's Docket No.: 10559-076002 / P7568C

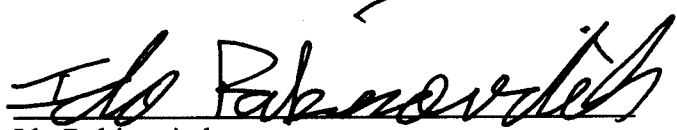
Any circumstance in which the applicant has (a) addressed certain comments of the examiner does not mean that the applicant concedes other comments of the examiner, (b) made arguments for the patentability of some claims does not mean that there are not other good reasons for patentability of those claims and other claims, or (c) amended or canceled a claim does not mean that the applicant concedes any of the examiner's positions with respect to that claim or other claims.

No fee is believed due. Please apply any other charges or credits to deposit account 06-1050, referencing attorney docket 10559-076002.

Respectfully submitted,

Date:

July 25, 2000



Ido Rabinovitch
Attorney for Intel Corporation
Reg. No. L0080

PTO Customer No. 20985
Fish & Richardson P.C.
Telephone: (617) 542-5070
Facsimile: (617) 542-8906